International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 22 (2015) pp 43256-43261

Dual Delegation Model – A Solution against Session Fixation Exploitation in Web Applications

A. Saravanan

Department of MCA, Sri Krishna College of Technology, Coimbatore, Tamil Nadu, INDIA a.saravanan21@gmail.com

M. S. Irfan Ahmed

Department of MCA, Nehru Institute of Engineering and Technology, Tamil Nadu, INDIA msirfan@gmail.com

S. Sathya Bama

Department of MCA, Sri Krishna College of Technology, Coimbatore, Tamil Nadu, INDIA ssathya21@gmail.com

Abstract

In this growing internet era, the number of web applications on the Internet is growing rapidly. Unluckily many of them have several security breaches. These security breaches can range from minor issues to even tragedies for the system and its owners. Thus the vast growth of information technology not only increases the number of web applications, but also increases the numbers of security vulnerabilities. Many web applications available today have methods for managing the sessions which bind with a particular user based on user authentication. However it is also possible for the adversary to exploit session management in order to impersonate the original user. Many solutions have been proposed over the years, which use either server side solutions or client side solution. This paper provides the information about 'Session Fixation Attack'. Also a framework has been proposed that uses a dual solution to detect and prevent Session Fixation vulnerability on the client and web server.

Keywords: Session Management, Session Fixation Attack, Web Applications, Prevention and Detection.

Introduction

The World Wide Web has evolved into a large dynamic interactive platform composed of millions of applications and services. In the beginning, the internet was used highly as information repository to store only static web pages and provides static information to the users. However, over the last few years, due to the rapid growth of internet, the web sites become more professional and dynamic. Thus, web applications are used to generate dynamic web pages and provide a way for implementing and providing access to on-line services. All communications between users via social networking and for almost all the activities of the user, web applications are available. Hence, Internet is becoming an integral part our daily life. Generally web applications are designed for user's convenience but they may be vulnerable to dangerous attack since they deal with users' sensitive personal information. The common vulnerabilities of web applications are SQL injection, cross site scripting, cross–site request forgery, security misconfiguration, broken authentication, session management and more. These vulnerabilities are listed in OWASP's Top 10 Project, a leading organization in the field of web application security.

Most web applications handle user authentication via the concept of web sessions. These allow users to use a web application without having to enter their login credentials for each and every action. Unfortunately, web sessions have many security weaknesses. OWASP, rates this weakness in web sessions as the second most important web application security risk [1]. Furthermore, many high-profile web applications are vulnerable to attacks on session management: YouTube and Twitter are two examples of web applications had such vulnerabilities in the past [2, 3]. If vulnerability occurs in very famous websites, a lot of visitors' information will be hacked and results in successful completion of harmful activities against the user, since many people transfer money using Internet banking, send credit card numbers to online shops and even more. Thus, mechanisms are needed to ensure security for internet users and to provide safe networking environment. Specifically, managing the web session is much more important to preserve the users' integrity and privacy over the internet [4].

An attacker can exploit security issues in session management mechanisms in order to impersonate a legitimate user on a website in several ways. Session Fixation Attack which is considered as a variant of session hijacking attack is one of the most common vulnerabilities in web applications. In this attack, the attacker fixes the user's session ID before the user even logs into the web server, thereby eliminating the need to obtain the user's session ID [5]. However, the users of a web application trust the developer and the application that it contains no security holes. Unfortunately, many times, the web developer might unaware about the security holes in their developed application. Therefore, the single solution on the server-side or on the client-side is not sufficient. To protect against a session attacks, a dual solution has to be implemented on client-side and on the server-side is indispensable.

The flow of the paper is as follows. The section 2 describes the need for web sessions. Next, section 3 points about the methods to manage session by transmitting the SID from the Client. This section explains about three methods and the

reason for why these methods are vulnerable to attack is also included. In section 4, Session Fixation attack and simple measures to mitigate the attack is pointed. Section 5 provides the literature survey. Section 6 explains the proposed work and its evaluation in detail. Finally Section 7 describes the conclusion and future work.

Need for Web Sessions

Unlike a single user application, the web application residing in a server has multiple clients. When a user visits a website, the web application often needs to remember for which user it is performing the actions. Thus, to identify actions of each user uniquely, the concept of 'Web Sessions' was invented. Web sessions are required for the stateful communication between the web server and the web browser. The browser requests web pages from a web server [6]. The server in turn provides the requested web page as a response. Unfortunately, HTTP [7, 8] is a stateless protocol, which means that the web server has no means to identify the two different requests from the same user. Because of this, a mechanism is needed for the HTTP to maintain the state of the user. Web sessions are such a mechanism which provides a way for stateful communication. The session is a key-value pair. Here the key is the session identifier and the value is the data about the user. Session Identifier (SID) is an alphanumerical value which is used to uniquely identify the corresponding Web session.

When the web server receives its first request from a particular client, it creates a session identifier (also called a session ID or SID) and associates the generated SID with the client. Also it sends the SID to the client as part of the response. In successive interactions, the client is instructed to include the assigned SID with every request, allowing the server to associate multiple requests to the same user using the same SID. Then the user tries to log in to the application by providing user name and the password as a request containing his SID. If the user name, password and the SID is authenticated, then the user is allowed to access the application. Thus SID is one of the user's authentication credential.

Session Management

Client can include the SID in their successive request using one among the three methods [9, 10]: 1) Cookies 2) URL arguments and 3) hidden form fields.

Among these three methods, cookies are the most convenient and most commonly used most method [11, 12, 5]. Cookies are strings containing the name-value pairs which are included in the Set-Cookie header of a HTTP response by the web server. Upon receiving a cookie, the browser stores it for a specified amount of time. The browser then attaches all known cookies for that domain to every successive request using the Cookie request header. This is done by including the cookie as the value for HTTP's Cookie header. Cookies can also be bound to a parent domain in which case they belong to all subdomains and it is used to share the same cookies among different parts of an application. Also, a new SID can easily be assigned by sending a new cookie with the same name but a different value to a client, which overwrites the old value [13].

The second method to transmit the SID is to include the SID as an argument in every request to the server. Since it alters the parameters in the URL, it is also called as *URL rewriting* [14]. In this case, the web server appends the session ID as a parameter with every URL that points to a page on the web server's own domain as a response. Thus, when the user clicks a link on the served page, the request that is made contains the session ID as a parameter, allowing the server to extract the SID and to associate the request with that session. The third possibility is very similar to URL rewriting, but uses POST instead of GET parameters [15]. Here, the session ID is included as a <form> element. When the user submits the form, the session ID is sent along with the request.

A. Sessions – Vulnerable to Attacks

There are some important things to note about session IDs. Session IDs are not only identification tokens, but also authenticators. Session IDs serve as temporary static passwords for accessing their sessions. Upon login, users are authenticated based on their credentials along with their issued SID. This makes session IDs a very appealing target for attackers. Moreover, there is no standardized way in session management. This means that different web applications will use different SID names, and that they will generate SID values in different ways. In many cases, an attacker tries to access the valid SID of the legitimate user. Several class of attacks focus on obtaining the session ID stored in browser's cookie storage. The attack in which the attacker gains access to the user's session by obtaining his session ID, is called *session hijacking* [16, 17].

Web session security is mainly focused on preventing three types of attacks against session IDs: interception, prediction and brute-force attacks. Encrypted communication effectively protects against interception [18, 15]. Using *cryptographic* pseudorandom number generators and carefully chosen seeds that don't leak from the server prevent prediction of session IDs. Finally, session IDs are immune to brute-force methods if their effective bit-length is large enough with respect to the number of simultaneous sessions [19]. Proposals have been made for mitigating the threat of stolen session IDs [20]. However, there is another type of attack called *fabrication*. Here the attacker will issue a session ID to the legitimate user and then force the user's browser to use the known session for login. After successful login of the target user, the attacker gains the control of the user's session using the known SID. This type of attack is called as *session fixation.*

Session Fixation

The goal of attacker in session hijacking and session fixation is to use the legitimate user's session and to mimic as user. However, in session fixation, instead of capturing the victim's session ID the attacker forces the victim to use a SID that is known in advance. The simple illustration for session fixation attack depicted in Fig.1 and the steps are as follows: [21, 5]:

1. The attacker opens the application and sends the request to the web server.

2. Web server generates and issues the SID 111 to the attacker as a response.
3. The attacker chooses the victim and forces the user to use the newly created session ID by providing the link to the server.
4. When the user clicks the link provided by the attacker, it takes to the application's login page that resides in the server. The request with the session ID 111 will be sent to the server. Awkwardly, the server accepts the already generated SID without creating the new one.
5. The user uses his credentials to log in at the server and the server grants him access to the application. Miserably, there exists a session at the server, identified by the SID 111 known to the attacker, in which the user is logged in.

By the meantime, the attacker also makes a request to the server, attaching the captured session ID 111 as his own. Thus, the attacker is able to impersonate the victim at the server.

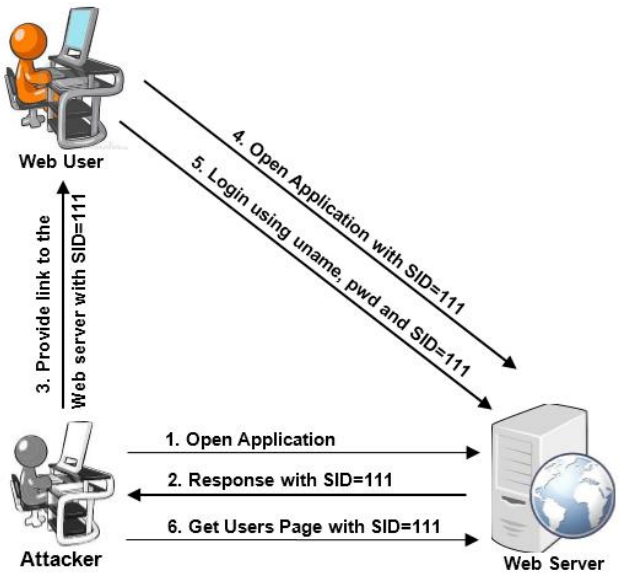

Fig.1. Architecture of Sensor Node

Some servers also accept crafted SIDs [22, 23, 24]. In this case, the attacker can just make up a new session ID, and no request needs to be made to the server. The server will adopt the SID sent by a user at his first request without generating the new one. Also the attacker uses several methods to inject the session ID in to the target user's browser. Session ID can be injected using URL parameters [25, 5], cross-site scripting [20], <meta> tag [5], header injection [26].

A. Countermeasures

Generally there are several security measures that can be implemented on the web applications against the session fixation vulnerability.

- Preventing logins to a chosen session
- Modifying the session ID after successful login
- Session ID can be generated based on the subnet address

Thus for web developers building a new web application, generally the best option is to use a web application framework that has session protection already built in. Alternatively, a server side proxy can be put in place to handle session security. Web application users also have some options to protect themselves. However, each method is lacking or they are not effective in mitigating attacks. Thus, a dual solution to detect and prevent session fixation attack on client and server side has been proposed.

Related Work

Session Fixation has received little attention in the past, mostly due to the vulnerability's severity. The basic fundamentals of the attack and the most obvious attack vectors are described in [5]. Furthermore, OWASP and WASC added articles about Session Fixation to their security knowledge bases [27, 28]. In paper [21] assessment has been made to know how wide spread the vulnerability pattern is in today's applications. It also explains the reliable protection mechanisms by fixing server side proxy at firewall to mitigate security problems. However, as the OWASP best practices guide on Web Application Firewalls (WAF) [29] states, 'current WAFs can only prevent Session Fixation if the WAF manages the sessions itself'. Furthermore, several protection techniques have been proposed that utilize proxies to mitigate related Web application vulnerabilities. In addition, Several other solutions has been proposed for other attacks like cross site scripting [30], Cross site request forgery attack [31], SSL Stripping attack [32].

Dual Delegation Model for Session Fixation Attack

In this proposed method a proxy will be implemented both in the client and server. The main advantage is the model can be added at the top of the existing architecture. Therefore, it is not necessary to modify the existing architecture. Also, the mechanism is not directly implemented in client and server system since it might become an additional overhead which reduces their performance. Instead the proxy could be added to the client and server as delegates and does the security checks and manage the sessions by monitoring the communications between client and the server. Since the proposed method uses dual proxies for the client and for the server separately, this model is named as dual delegation model. The framework for the proposed Dual Delegation Model is depicted in Fig.2.

Both the server proxy and the client proxy contains a Session ID database. All the incoming session ID will be compared with the values in the database. The data will be forwarded to client and server only if it is valid. Initially the client sends a first request to the client proxy which forwards to server proxy. Since there is no session ID associated with this request, it will be forwarded to the server. The server then generates the session ID and sends this as a response through server proxy. The server proxy generates a login ID for the request, associates this with SID and stores both in its

database. Also it forwards it to the client. The client receives the ID pair, stores this value in its database for further communication and then allow it to reach the client by removing the Login ID. Whenever the client sends a request with the session ID, the client proxy will check the value in its database. The client proxy will allow the request only if the SID matches with the one in the database. Thus it does not allow the first request with a session ID. Also before forwarding the request, it appends the current Login ID that matches with the SID in the database. Similarly the server proxy will check for the SID and Login ID pair in the database. It will forward only it the two IDs matches with the one in the database.

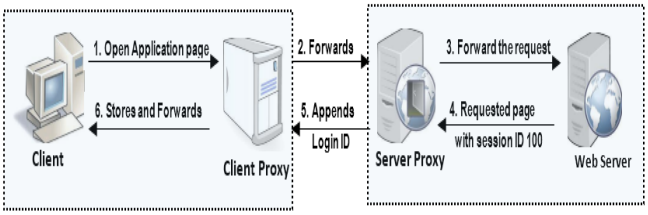

Fig.2. Dual Delegation Model Framework with an Illustration

Another important thing is that it changes the Login ID each time when the user login is made. Thus even the session ID and Login ID is known to the attacker during authentication of the user, after successful login the Login ID will be regenerated which is not known to the attacker. Hence, the session's security does no longer lie in the SID of the vulnerable application but in an additional identifier issued by the proxy. Accordingly this proxy can be put in front of almost any vulnerable application, giving the application's developers time to fix it.

The temporary Login ID is similar to Proxy Session Identifier (PID) mentioned in [33]. However, successful and intellectual creation of Login ID could even protect Session Hijacking attack. If the Login ID is generated based on the allocated SID and unique address of the user, then even the session is hijacked, and even the attacker request the page with the hijacked session ID, the unique address would not be same for attacker and the user and thus the request will be the discarded. Thus each time when the server proxy receives request with SID, it calculate Login ID and it checks with the value stored in the database for that SID. The request will be allowed only if there is a match. Thus, the Login ID can be generated and verified by using simple XOR operation to improve the performance.

A. Evaluation

Simple evaluation has been made by creating the HTTP server to make sure that the implementation was according to the specification. This server issues different kinds of cookies (both via HTTP and via JavaScript), and checks which cookies are returned over HTTP to see what delays were introduced in a normal browsing session. This was done separately for the requests (where it had to be checked whether a cookie would be allowed) and the responses (where new cookies could be set). The evaluation was done on a computer with a dual-core processor running at 1.7GHz. The real-world evaluation has to be made to check the performance of the web server with several request. In recent years receiver operating characteristics (ROC) graphs are commonly used in medical decision making, machine learning and data mining. It is a technic for visualizing and classifying the performance of the system [34]. ROC classification models are based on mapping elements into two different set of classes. Here the first set is used to denote the originality of the data. The attacks are denoted as positive condition *P* and user data is denoted as negative condition *N*. The second set is the classification made by the proposed system. If the system predicts the data as attack, it is denoted by true *T* and it predicts the data as user data, it is denoted by false *F*. Given an instance is positive and also classified as true it gets counted as true positive (TP); if the instance is instead positive it is classified as false it gets counted as false positive (FP). Similarly false negative (FN) and true negative (TN) can be evaluated. In this test, 50 instances were used as classifiers for evaluating predictions made by the proposed system.

Based on this the precision and accuracy have been calculated. Precision is how many of the predicted attacks are relevant to the decision maker and the formula is mentioned in eq (1). Accuracy is the overall correctness of the predictions made, the formula is mentioned in eq (2).

$$precision = \frac{TP}{TP + FP} \quad (1)$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (2)$$

The comparison of the proposed method has been made with the existing method given in [14]. The precision and accuracy of the proposed and existing method as a graphical representation in shown in the Fig 3 and Fig 4.

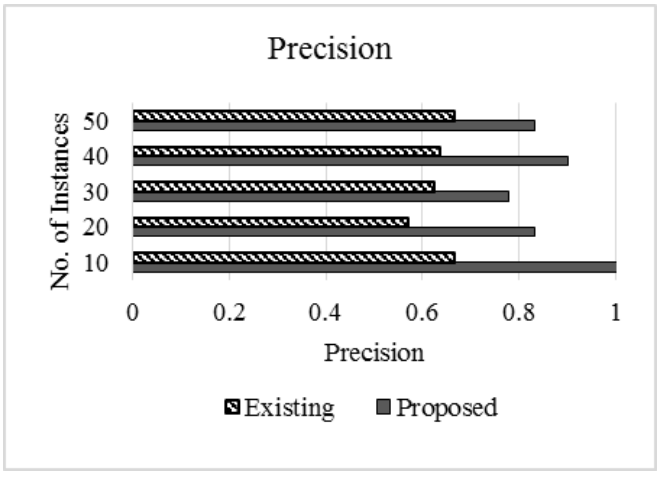

Fig.3. Comparison using Precision

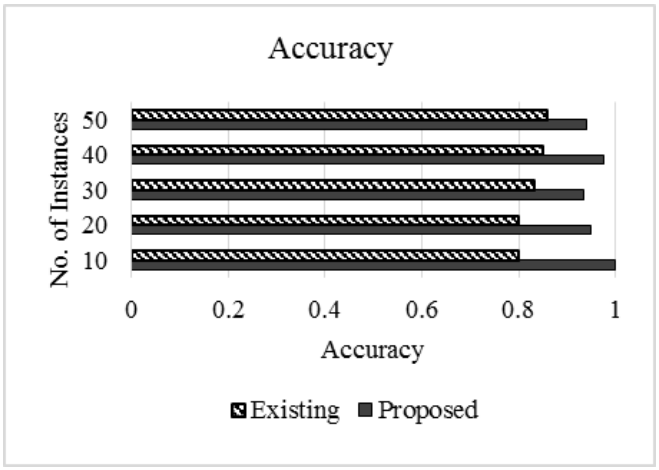

Fig.4. Comparison using Accuracy

Conclusion

In this paper, we discussed in detail about the web sessions and why they are important for internet communication. Also, we discussed about the reason for the vulnerability in web sessions and its implementations. We studied the different attack classes and in particular a thorough study about session Fixation attack has been made. In this research, we proposed a dual delegates framework to find the session fixation attack and it can be extended to prevent from session hijacking attack. It separates and inserts the proxies for client and server. A simple web server and client has been implemented to check for the proposed method. Thus, it requires minimal effort and it is robust to attacks. We extend our work to find the solution for other web session vulnerability and could be extended for Cross site scripting attack.

References

[1] "OWASP top 10 vulnerabilities".
[2] Stefan Tanase. "Twitter XSS in the wild", 2007.
[3] Bojan Zdrnja. "Stored XSS vulnerability on YouTube actively abused?" SANS Institute: Internet Storm Center Diary, 2010.
[4] Mariusz Stawowski, "Client side Vulnerability Assessment", retrieved from
[5] Kolšek, Mitja. "Session fixation vulnerability in web-based applications", Technical Report, Acros Security, 2002.
[6] James F. Kurose and Keith W. Ross. "Computer networking: a topdown approach." Pearson/Addison-Wesley, 4th edition, 2008.
[7] IETF, "RFC2616: Hypertext Transfer Protocol -- HTTP/1.1".
[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners Lee. "Hypertext Transfer Protocol – HTTP/1.1. RFC 2616".
[9] IETF, "RFC2109: HTTP State Management Mechanism" http://www.ietf.org/rfc/rfc2109.txt
[10] Nenad Jovanovic, Engin Kirda, and Christopher Kruegel, "Preventing cross site request forgery attacks", In Securecomm and Workshops, 2006, pages 1–10. IEEE, August 2007.
[11] David M. Kristol. "HTTP Cookies: Standards, privacy, and politics", ACM Transactions on Internet Technology (TOIT), 1(2):151–198, November 2001.
[12] Joon S. Park and Ravi Sandhu. "Secure cookies on the Web", IEEE Internet Computing, 4(4):36–44, 2000.
[13] De Ryck, Philippe, Nick Nikiforakis, Lieven Desmet, Frank Piessens, and Wouter Joosen. "Serene: Self-reliant client-side protection against session fixation", In Distributed Applications and Interoperable Systems, pp. 59-72. Springer Berlin Heidelberg, 2012.
[14] Bonné, Bram. "Improving session security in web applications", 2011.
[15] Martin Johns. "SessionSafe: Implementing XSS immune session handling", Computer Security–ESORICS 2006, pages 444–460, 2006.
[16] The Open Web Application Security Project, "Cross Site Scripting".
[17] The Open Web Application Security Project, "Session Hijacking".
[18] ACROS, "Remote Retrieval of IIS Session Cookies from Web Browsers".
[19] David Endler, "Brute-Force Exploitation of Web Application Session IDs".
[20] Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster, "Dos and Don'ts of Client Authentication on the Web".
[21] Martin Johns, Bastian Braun, Michael Schrank, and Joachim Posegga. "Reliable Protection against Session Fixation Attacks", In SAC '11 Proceedings of the 2011 ACM Symposium on Applied Computing, pages 1531–1537, New York, New York, USA, 2011. ACM.
[22] Chris Shiflett. "Session Fixation", 2004.
[23] Chris Shiflett. "Session Hijacking", 2004.
[24] Martin Johns and Justus Winter. "RequestRodeo: client side protection against session riding", In Proceedings of the OWASP Europe 2006 Conference, refereed papers track, Report CW448, pages 5–17, 2006.
[25] Will Bontrager. "Form Submissions Without Submit Buttons", 2005.
[26] Amit Klein. "Divide and Conquer" - HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics. Technical report, Sanctum, Inc., January 2004.
[27] The Open Web Application Security Project (OWASP). "Session Fixation".
[28] The Web Application Security Consortium (WASC). "Session Fixation".
[29] OWASP German Chapter. "OWASP Best Practices: Use of Web Application Firewalls". [white paper], July 2008.
[30] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic. "Noxes: A Client-Side Solution for Mitigating Cross

Site Scripting Attacks". In 21st ACM Symposium on Applied Computing (SAC 2006), April 2006.

[31] N. Jovanovic, C. Kruegel, and E. Kirda. "Preventing cross site request forgery attacks", in Proceedings of the IEEE International Conference on Security and Privacy for Emerging Areas in Communication Networks (Securecomm 2006), 2006.

[32] N. Nikiforakis, Y. Younan, and W. Joosen. Hproxy: "Client-side detection of SSL stripping attacks", in Seventh Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'10), 2010.

[33] M. Schrank, B. Braun, M. Johns, and J. Posegga. "Session Fixation - the Forgotten Vulnerability?" in Proceedings of GI Sicherheit 2010, Lecture Notes in Informatics (LNI), 2010.

[34] Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861-874. doi:10.1016/j.patrec.2005.10.010.